Using Explicit Requirements and Metrics for Interface Agent User Model Correction

Proceedings of the Second International Conference on Autonomous Agents (Agents '98), pp. 1-7

10-13 May 1998

Scott M. Brown, Eugene Santos Jr., Sheila B. Banks, and Mark Oxley

Department of Electrical & Computer Engineering
Air Force Institute of Technology
2950 P Street, Wright-Patterson AFB, OH 45433-7765

Using Explicit Requirements and Metrics for Interface Agent User Model Correction

Scott M. Brown
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433–7765 USA
sbrown@afit.af.mil

Eugene Santos Jr.
Computer Science and Engineering
University of Connecticut
Storrs, CT 06269–3155 USA
eugene@eng2.uconn.edu

Sheila B. Banks[†] Mark E. Oxley[‡]
[†] Department of Electrical and Computer Engineering
[‡] Department of Mathematics and Statistics
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433–7765 USA
{sbanks, moxley}@afit.af.mil

Abstract

The complexity of current computer systems and software warrants research into methods to decrease the cognitive load on users. Determining how to get the right information into the right form with the right tool at the right time has become a monumental task — one necessitating intelligent interface agents with the ability to predict the users' needs and intent.

An accurate user model is considered necessary for effective prediction of user intent. Methods for maintaining accurate user models is the main thrust of this paper. We describe an approach for dynamically correcting an interface agent's user model based on utility theory. We explicitly take into account an agent's requirements and metrics for measuring the agent's effectiveness of meeting those requirements. Using these requirements and metrics, we develop a requirements utility function that determines *when* a user model should be corrected and *how*. We present a correction model based on a multi-agent bidding process and the aforementioned metrics and utility function. Finally, we discuss several critical research issues concerning the use of user models that open fertile ground for future research.

1 Introduction

Today's computer systems and software are more accessible to users due to reduced cost, increased necessity, and technological advances than they have ever been. Processing the sheer volume of information and dealing with the increased complexity of software and computer systems has become progressively difficult. Automatically determining how to get the right information into the right form using the right tool at the right time has become a necessary and monumental task for today's computer systems.

In an attempt to address the complexity issues of current software and information processing, software agents have been steadily finding their way into various applications. Research into one type of software agents, termed *interface* or *user agents*, in complex systems to carry out actions on the user's behalf is still in its infancy. Previous systems using agents [19] have fallen short of providing adaptive, collaborative, robust, and autonomous agents. Most of these agents have been either pedagogical (e.g., modeling a small fraction of the possible actions) or narrowly focused (i.e., the results are useful for a very small number of architectures).

One underlying problem of interface agent research is the failure to adequately address effective and efficient knowledge representations suitable for modeling the users' interactions with the system. Current interface agents lack the representational complexity to manage the uncertainty and dynamics involved in predicting user intent and modeling user behavior. Our research focuses on these issues.

The organization of this paper is as follows: we first provide the motivation and background information for our current research efforts in Section 2. Section 3 outlines requirements we believe interface agents must meet, metrics to measure those requirements, and a methodology for determining if an agent is meeting those requirements. In Section 4, we discuss our approach to correcting an interface agent that is not meeting its requirements by modifying the underlying user model using a multi-agent bidding process. Finally, we draw conclusions from our research and show promising areas for future research.

2 Background

Interface agents are a relatively new area of research within the artificial intelligence (AI), human-computer interaction (HCI), and user modeling communities[1]. If we had to choose one word to characterize both the AI and HCI communities' research into interface agents, the words would be "delegation" and "customization", respectively. The AI community as a whole has concerned itself with what an interface agent can do *for* the user, whereas the HCI community has

[1]While we do not mean to stereotype a particular research field too narrowly, we feel it is advantageous to point out the general "slant" different research communities take on interface agent research.

To appear in the *Second International Conference on Autonomous Agents (Agents '98)*, May 10–13, 1998.

concerned itself with what the user can do *with* the agent. The strength of AI research lies in its years of experience in knowledge representation, reasoning, and machine learning. The strength of HCI research in interface agents is its attentiveness to the user, focusing on what a user needs to perform his/her tasks, and how best to represent information to the user. While there is no clear delineation between the two groups with regards to their research in interface agents, both approach the research field of interface agents differently.

The field of user modeling is perhaps a good marriage between artificial intelligence and human-computer interaction. User modeling is concerned with how to represent the user's knowledge and interaction within a system to adapt those systems to the needs of users. Researchers from the fields of artificial intelligence, human-computer interaction, psychology, education, and others have all investigated ways to construct, maintain, and exploit user models. Employment of user models increases effectiveness and/or usability of systems implementing the various techniques from the user modeling field [18]. In recent years, a number of systems have used numerical uncertainty techniques from the AI community to capture the uncertainty inherent in modeling users [17].

Our own research in the field of intelligent interface agents is demonstrated by our Core Interface Agent Architecture (CIaA)[2] [7], integrated into an expert system shell called PESKI [12, 13]. PESKI (Probabilities, Expert Systems, Knowledge, and Inference) is an integrated probabilistic knowledge-based expert system shell. PESKI provides users with knowledge acquisition [27], verification and validation [4, 28], data mining [31], and inference engine tools [29], each capable of operating in various communication modes. PESKI was used for our initial tests concerning implementation and usability of our intelligent interface agent [2]. Within the PESKI domain, CIaA has been used to help the user select the various tools in a given communication mode ("standard" windows, icons, menus, and pointers direct manipulation interface or a graphical, drawing tool-based interface) given the current context of interaction. For our initial tests, the CIaA based its user model on user class stereotypes [26], a causal planning model (represented as a Bayesian network [24]) of the users' interaction with PESKI [17], and the statistical frequency of interaction with the various tools and communication modes. We are currently expanding the CIaA user model to cover more aspects of the PESKI domain. For more information on PESKI, see the Air Force Institute of Technology's Artificial Intelligence Laboratory web site[3].

CIaA supports effective user intent prediction by incorporating the ability to model both the uncertainty in user intent and dynamic user behavior within its user model [7]. To effectively predict user intent, an accurate cognitive model of the user is considered to be necessary. The problem with most cognitive models for intelligent interface agents is they rely on knowledge representations lacking flexibility and power in two key areas: the representation of uncertainty and dynamic user modeling. Employing a knowledge representation that correctly captures and models uncertainty in human-computer interaction can improve the modeling of the user and the user interface behavior. One representation that is ideal for representing uncertainty is Bayesian Networks (BNs) [24]. Bayesian techniques have attractive properties for developing interface intelligence because they capture uncertainty, required to model user intent. Also, Bayesian techniques are extremely useful in predicting future events. The dynamics of user modeling may be captured via user preference random variables in the Bayesian network [5, 6] as well as the use of dynamic Bayesian networks that change structure over time [14].

[2]CIaA was formerly known as the Intelligent Interface Agent (IIA).

[3]`http://www.afit.af.mil/Schools/EN/ENG/LABS/AI/`

2.1 Symbiotic Information Reasoning and Decision Support

The driving goal of our research is to develop a comprehensive software engineering, knowledge engineering, and knowledge acquisition methodology for Symbiotic Information Reasoning and Decision Support (SIRDS). The underlying idea of SIRDS is to allow the user to perform the tasks he/she can do well, and the agent to perform the tasks it does well. Agents' strength lies in their ability to perform data acquisition and management (to include display of this information [16]) from many heterogeneous sources, low level quantitative and qualitative data analysis, and routine inference to enable decision support. The user's strength lies in his/her ability to provide guidance and insight into the information that is necessary to draw complex, higher level inferences from the data [3]. A symbiotic approach is necessary because the objective is to let the user and the computer share the task load; therefore, we use a human-centered approach to task partitioning.

SIRDS requires the development of an adaptive, intelligent, learning human computer interface. Intelligent agents are a key aspect of SIRDS, and they perform information fusion, analysis, and abstraction, as well as deriving information requirements and controlling information display. These agents within SIRDS are necessarily of two different types, one type for the task of reasoning to direct system data acquisition, assessment, and information synthesis; and the other for reasoning about information display. However, the same software architecture and development methodology is employed to realize both types of agents. Our CIaA is a first step towards realizing our research goal.

3 Interface Agent Requirements and Metrics

For the agent to perform within its environment, we must determine *what is important* to model in the domain, with associated discriminators and/or metrics to determine and define *when* and *how* to dynamically change the user model. In previous evaluations we identified potential problems with our interface agent's user model [3]. To improve an interface agent's utility for providing timely, beneficial assistance to the user, we believe agent development must explicitly take into account the agent's requirements. Methods for determining when those requirements are not being met and how to correct the agent's user model is the primary focus of this paper.

Many of the requirements of agents are not well defined and many of these requirements are not mutually exclusive. Furthermore, these requirements do not set agents apart from other forms of software (see Petrie [25]). This shortfall has apparently caused most researchers to ignore requirements for agent development. We explicitly consider requirements for interface agents. In this section, we develop a concrete (i.e., well defined), measurable set of requirements, with associated requirement metrics, to determine if our interface agent is meeting the requirements. We categorize each metric under the top-level requirement associated with it; however, a clear delineation between the requirements is not always possible and therefore some of the metrics measure more than one requirement. Each metric is evaluated on a scale from 0 to 1, 1 signifying the agent perfectly meets the requirement.

We present the following four requirements we feel best capture the essence of interface agents as they exist today. Due to space limitations, we present a subset of the metrics associated with each requirement used to determine if the interface agent is meeting the requirement.

Adaptivity — We define adaptivity as "the ability to modify an internal representation of the environment through sensing of the environment in order to change future sensing, acting, and reacting for the purpose of determining user intent and improving assistance." Adaptivity implies a number of sub-requirements. It implies an agent is *perceptive*. That is, the agent can distinguish relevant features in the environment in relationship to the method necessary to act and react within the environment. Concerning an agent's ability to act and react, the adaptivity requirement assumes the agent is *capable* to affect the environment through its acting and reacting. *Reactive* "behavior" implies a timely response (i.e., stimulus-response) to sensed events, whereas to *act* connotes a deliberative (reasoned) response to events. The relationship between the agent's sensing, acting, and reacting with regards to the internal representation of the environment yet further define properties for the agent. Goodwin [11] defines these deliberative agent properties to include *predictive* — the ability to model the environment so as to predict how its actions will affect the environment, *interpretive* — the ability to correctly assess its sensors, and *rational* — the ability to perform actions to obtain its goals.

The *precision metric* measures the interface agent's ability to accurately suggest assistance to the user. We define our precision metric as

$$M_{precision} \triangleq \frac{number\ of\ correct\ suggestions}{number\ of\ suggestions}. \qquad (1)$$

The precision metric does not account for the agent's inability to suggest assistance. That is, if an agent is unable to suggest assistance (e.g., due to a slow reaction time), this metric will not measure this. To measure this aspect, we can define an assistance capability metric, measuring the agent's *capability* to offer suggestions, as defined by the percentage of times an agent is able to suggest correct assistance based on a particular "state of the world" before the state of the world changes (e.g., the agent receives more information from the application about the environment).

This metric, in effect, measures the ability to provide timely assistance to the user and is defined as

$$M_{assistance\_capability} \triangleq \frac{number\ of\ correct\ suggestions}{number\ of\ state\ of\ world\ changes}. \qquad (2)$$

The precision and assistance capability metrics are related to the precision and recall statics, respectively, used in text filtering systems [23] where the precision statistic is defined as a ratio of the number of relevant found documents to the total number of documents found and the recall statistic is defined as the fraction of the actual set of relevant documents that are correctly classified as relevant.

The *reactive metric* measures how quickly the agent can respond (i.e., act) to an environmental stimulus. We define $T_{sense \rightarrow act}$ as the measure from the time the stimulus is sensed (i.e., received) by the agent to the time the agent is able to act. Note that no explicit action is considered an action. For example, the agent may determine the user does not need assistance currently. To scale the metric, we establish a cutoff time, T_{cutoff}, that the agent must respond within. We define this metric as

$$M_{reactive} \triangleq 1 - \frac{T_{sense \rightarrow act}}{T_{cutoff}} \qquad (3)$$

It is possible, and may be necessary, to refine the reactive metric's granularity. That is, we may need to decompose the metric into various sub-metrics, each measuring a portion of $M_{reactive}$. Then, we can determine possible "bottlenecks".

Autonomy — This requirement, more than any other, seems to define agency as Petrie [25] argues. However, he also notes autonomy is not well defined within the community. Franklin and Graesser [9] argue autonomy implies a reactive (sensing and acting within a time constraint), temporally continuous, and goal-oriented (pro-active) agent. Based on the research conducted to date, we define it as "the ability to sense, act, and react over time within an environment without direct intervention." By direct intervention, we mean explicit "activation" by the user or other agents of the agent. This definition does not preclude our agent from responding to this sort of collaborative interaction with the user and other agents. Instead, we desire our agent to be able to act on behalf of the user without being "told" to do so. Indirect intervention takes the form of "looking over the shoulder" [20] of the user to determine what the user is doing and determining how to assist the user. Autonomy implies some sort of saved internal "state", whereas adaptivity explicitly requires it. One external measurement is to measure the number of suggestions the agent offers to the user without the user's request. We define this metric as follows:

$$M_{external\_autonomy} \triangleq \frac{number\ of\ autonomous\ suggestions}{number\ of\ suggestions}. \qquad (4)$$

Collaboration — We require all interface agents to collaborate with a user. Collaboration with the user best differentiates interface agents from other types of agents. This collaboration may be as simple as making a suggestion to the user and asking if the suggestion was correct or not, or as complicated as observing the user's actions within the environment, attempting to determine the needs and intent of the user, and providing assistance at "appropriate" times. Collaboration allows agents to increase their internal representation accuracy, resolve conflicts and inconsistencies within the representation, and improve their decision support capabilities [32]. Collaboration may also be with other non-human agents. For heterogenous agents, collaboration implies an agreed upon agent communication language and a commitment to use that language. Central to collaboration are the *behaviors* an agent can perform and the *protocol* in which they communicate those behaviors [8]. Based on this discussion, we define interface agent collaboration as "the ability to communicate with other agents, including the user, to pursue the goal of offering assistance to the user."

Collaboration metrics are therefore used to measure both collaboration with the user and other agents — in our case, correction adaptation agents (discussed in the next section). With regards to measuring the collaboration requirement with correction adaptation agents, we are concerned with the efficiency of the collaboration using the agent's communication language[4]. There is a plethora of performance met-

[4]We are currently using KQML [21].

rics for communication protocols [30] we can use to determine the ability to communicate with other agents over networks. While although not all the factors determining these metrics are within the control of the interface agent (e.g., network throughput is largely determined by the bit rate), knowledge of communication bottlenecks can help the interface agent make informed decisions about accessible and applicable information sources. With regards to measuring the interface agent's ability to collaborate with the user, we are concerned with the effectiveness of the collaboration. We define a metric to measure the level of collaboration. First, we define a metric to measure the percentage of the "workload" the agent performs versus the user as follows:

$$Agent_{workload} \triangleq \frac{\#\ of\ agent\ actions}{\#\ of\ agent\ actions + \#\ of\ user\ actions}. \quad (5)$$

Note that we base this metric on actions and not suggestions since a suggestion may be made of several actions. Each user can determine the amount of collaboration he/she desires by setting a *collaboration threshold*, ranging from 0 to 1, specifying the desired amount of collaboration between the interface agent and user. For example, a collaboration threshold value of 0.5 means the user desires the "workload" to be divided evenly between the agent and user, whereas a threshold of 0 indicates the user wants the agent to perform no actions on his/her behalf. Using this threshold, we can define the *collaboration metric* as follows:

$$M_{collaboration} \triangleq 1 - |T_C - Agent_{workload}|, \quad (6)$$

where T_C is the user's collaboration threshold.

Robustness — We add this requirement for interface agents since they, more than other agents, must be capable of gracefully degraded performance. We require this capability of interface agents because they must interact with the most complicated of agents — users. However, robustness is not limited to performance. Extensibility and maintainability are also key. We desire the ability to easily adapt the agent to information and requirement changes. The former is best dealt with under the adaptivity requirement (e.g., the ability to adapt to different users), while the latter is better dealt with during agent specification. The agent should have the ability to, for example, add new sensor information dynamically or be used in a new environment with different requirements. Therefore, robustness is "the ability to degrade assistance gracefully." The ability of the interface agent to correct its user model is partially related to the number of correction adaptation agent responses received. This correction process (described in detail in the next section) relies on the interface agent sending a "bid" to the correction adaptation agents, and they in turn responding within a time limit, $T_{cutoff_{CA}}$. Assuming N_{CA} correction adaptation agents, we define this metric as follows:

$$M_{response\_quantity} \triangleq \frac{number\ of\ CA\ agents\ responding}{N_{CA}}. \quad (7)$$

Similarly, we measure the average time it takes correction adaptation agents to respond:

$$M_{response\_time} \triangleq 1 - \frac{\frac{\sum^{N_{CA}} T_{CA_i}}{N_{CA}}}{T_{cutoff_{CA}}}. \quad (8)$$

3.1 Requirements Utility Function

We define a successful agent as "an agent with the ability to provide timely, beneficial assistance (suggestions, tutoring, help, interface adaptations)." This definition appears very open-ended. We attempt to close the ends by looking at the agent's utility in meeting these requirements.

The utility function $U_{requirements}$ is defined for the requirement metrics of the agent, weighted with respect to their importance, based on some previous history. That is,

$$U_{requirements} : \omega^n \times R^n \times H \mapsto \Re, \quad (9)$$

where for each history $h \in H$ of previous actions and events, $\omega \in [0, 1]$ is a weighting factor for each of the n requirement metrics R, and the utility function maps to a real number. ω can be a function of time, where we allow the weights to change depending on the current situation. For example, if the interface agent is making poor suggestions, we can increase the weight(s) associated with the adaptivity requirement metrics, denoting its increased importance. The higher the value of the utility of our interface agent, the more "successful" it is in meeting its requirements.

Our metrics do *not* explicitly take into account the recency of the actions determining the metrics. As presented, each metric is calculated over the entire history of actions and events. However, in practice, we may desire to have a fading function to weigh more heavily recent actions/events. Note, however, our utility function $U_{requirements}$ does take history into account. If we desire to evaluate the metric for the last ten events, for example, we limit our history to the last ten events and evaluate our metrics over this history. We may desire to have metrics evaluated over several different history lengths[5].

4 The Correction Model

As mentioned previously, an accurate user model is considered necessary for effective prediction of user intent. However, there are many possible causes for an inaccurate user model. The plethora of machine-learning techniques used in current user models attests to two simple facts: users are different and users' behavior changes over time. Both of these facts are causes of the same effect: a deviation of user behavior from the originally specified and designed user model (assuming we accurately captured the user's model initially). User models that fail to dynamically adapt to different users and the changing behaviors and/or needs of a user are doomed to be inaccurate in short order.

To account for the uncertainty and dynamics involved in predicting user intent and modeling user behavior, our set of requirement metrics and the requirements utility function can be used to determine when and how to correct the interface agent's user model. This section addresses *when* and *how* to dynamically change the user model and discusses how a set of correction adaptation agents can use the aforementioned requirement metrics, requirements utility function, and previous interface agent behavior to suggest changes to the user model. Research done previously focused on identifying the types of problems that may arise in an interface agent's knowledge representation [6]. These problems are a result of the interface agent failing to meet its requirements and indicate the agent's user model is inadequate to deal with the dynamic environment. Since we have a set of

[5]The advantage to evaluating metrics over varying history lengths is akin to stating mutual fund performance over 5, 10, and 20 years performance.

requirement metrics and an associated requirements utility function, we can readily identify which requirements are not being met and attempt to correct the problem by altering the user model.

Our approach to correcting the user model is to have the interface agent request "help" from correction adaptation agents — special agents capable of correcting problems with the user model by adapting it to improve the interface agent's requirements utility. We discuss a scenario for having a correction adaptation agent suggest changes to the ailing interface agent user model, based on the concept of a contractual bidding process. These agents engage in a "bidding process" to recommend changes to the ailing interface agent user model. The interface agent serves as a *manager* agent, responsible for determining when a contract is available (in our case, when the requirements utility function value is below a chosen threshold), announcing the contract to be filled, receiving bids from the *bidder* agents, evaluating the *utility* of the bids, and finally accepting or rejecting the bids based on their utility. The correction adaptation agent who may improve the interface agent's requirements utility the most "wins" the contract to correct the user model.

Two points need to be addressed. First, the use of correction adaptation agents to suggest adaptations to the interface agent's user model may not seem intuitive. It may appear the interface agent itself is better equipped to correct its own user model. However, as mentioned previously, our agent already possesses a number of ways to adapt to the dynamic environment. If the agent's requirements utility falls below the threshold, this indicates its existing adaptation mechanisms are incapable of dealing with the dynamic environment and needs specialized "attention". Second, the use of correction adaptation agents allows the interface agent to continue to observe the environment and possibly still offer assistance without utilizing computational resources correcting the user model.

We present our bidding process model adopted from Müller [22] as the underlying meta-level learning mechanism for our interface agent. The model is formally defined as follows:

Definition: Let D be our domain. Our model can then be represented as a 5-tuple

$$(A,\ N,\ U,\ \Pi,\ \Sigma)$$

where

- $A = \{a_1,\ \ldots\ , a_k\}$, $k \geq 2$, is a set of agents a_i with mental states $B_i = (O, h, E)$, where
    - $O \subseteq D$ is a domain ontology.
    - $h \in H$ is a history of interface agent suggestions and any information the agent used to make its suggestionsand user choices based on the assistance offered.
    - E is the effectiveness of the agent, with respect to the various requirement metrics.
- $N \subseteq D$ is the negotiation set and semantically represents the changes a bidder agent proposes to make to the user model.
- $U = \{u_1,\ \ldots\ , u_k\}$, where $u_i : A \mapsto \Re$ is a utility function for correction adaptation agent a_i, based on the interface agent's requirements utility function over time[6].
- $\Pi = (K, \pi)$ is a negotiation protocol, where
    - $K = \{start, done, ANNOUNCE, BID, REJECT, GRANT, REPORT\}$ represents the communication primitives.
    - $\pi : R \times K \mapsto 2^K$ is a protocol function mapping communication primitives for agent roles to allowable reactions.
- $\Sigma = \{\sigma_1,\ \ldots\ , \sigma_k\}$ is a set of negotiation strategies where $\sigma_i : \Pi \times A \times K \times 2^D \times U \mapsto K \times N$. Specifically, $\sigma_i(\Pi, a_i, k, N, u_i) = (k', N')$ with $k' \in \pi(a_i, k)$, $N' \subseteq N$.

[6]As mentioned previously, the weights for each requirement metric used to calculate the requirements utility function may change over time, and therefore, we store the weights in the history H for use of calculating the correction adaptation agents utility over the history.

The correction adaptation agents' architecture is very similar to the interface agent's [7], where each correction adaptation agent possesses its own user model, an evaluator for calculating the metrics and requirements utility function, and communication protocol handler for communicating with other agents. Additionally, each correction adaptation agent has a specialized bidding component capable of adapting the user model. However, the complete description of the architecture of the correction adaptation agents is beyond the scope of this paper. In practice, each correction adaptation agent maintains a user model that is identical to the interface agent's user model until the interface agent requests help from the correction adaptation agents, at which time each correction adaptation agent adapts its own user model based on its bidding behavior component.

The manager evaluates the utility by making the proposed changes to the "oldest" user model stored in the history H. Then, for each suggestion made to the user, the manager determines what the new suggestion(s) would be, based on the bidder agent's proposed changes and any evidence stored in the history. The requirements utility function value is then recalculated. The "winning" correction adaptation agent is the agent improving the requirement utility function by the greatest magnitude. We can bias this evaluation by multiplying the bidder agent's effectiveness E and the requirements utility function. The correction adaptation agents' effectiveness E is updated by simple reinforcement learning, where the "winning agent" receives positive learning [14]. The reinforcement learning takes into account those bidder agents determined to be "helpful." In practice, certain correction adaptation agents are more apt to improve certain metrics than other metrics. That is, the correction adaptation agent's bidding behavior component affects only certain metrics. Therefore, if a particular metric has a significantly greater weight, ω, than the other metrics and/or if this metric's value is significantly lower than the other metrics, we are likely to choose those correction adaptation agents capable of increasing the metrics value, thus increasing the requirements utility function's value.

The actual definition of Π and Σ is more appropriately left for the architecture. We can adopt any number of bid strategies. Currently, the negotiation protocol function, Π, and negotiation strategy, Σ, are defined generally the same as Müller [22], which is a sealed-bid, single award strategy. That is, the other agents have no idea what the "price" (i.e., utility) other agents are bidding. Other strategies include agents learning from past bids to improve future bids [33].

4.1 Analysis

Figure 4.1 shows the value of the interface agent's requirements utility function over a typical run with PESKI. For this example, we used an additive requirements utility function with equal weights for each of the requirement metrics. A threshold $T_{U_{requirements}}$ was chosen based on previous empirical results.

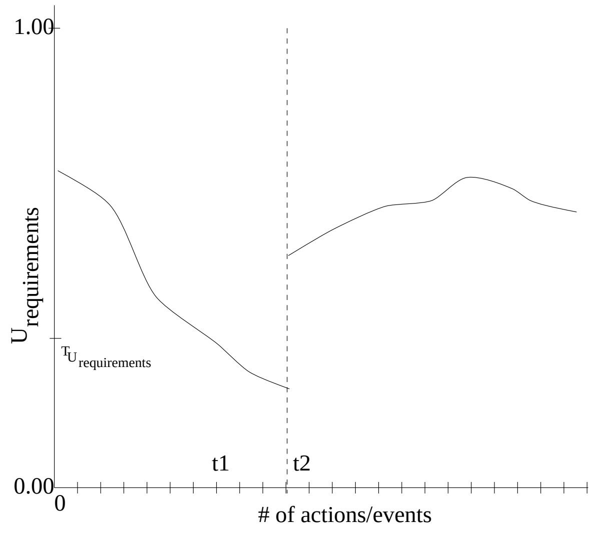

FIG. 4.1. Requirements Utility Function for the Interface Agent.

At time t_1, the agent's requirements utility function drops below the threshold and a contract announcement is made. Bids are made by the correction adaptation agents. All bids are received by time t_2 and the interface agent determines the correction adaptation agent with the best bid (i.e., highest utility) and "awards" the contract by allowing this agent to effect its changes to the user model. Since the correction adaptation agent's changes increase the requirements utility function, we see a discontinuity at t_2.

5 Issues, Future Research, and Conclusion

In this paper, we described our approach for dynamically correcting an interface agent's user model based on the agent's requirements. Our requirements utility function determines *when* a user model should be corrected and *how*. The development of a set of concrete, measurable requirements for our interface agent is uncharacteristically precise. In our observations, few researchers explicitly specify their agent's requirements and provide ways to measure those requirements. In stark contrast, we have stated our requirements, provided a way to measure the requirements, and use the requirements to measure our interface agent's utility for providing assistance to a user. Furthermore, our approach allows the specific relevancy of a requirement to change over time, as certain requirements become more or less important. How to change a user model is just as important, if not more so, than when to change the model. Is it better to fail to recognize a user model is no longer adequate than to change a user model incorrectly? Both problems have the same effect — an incorrect user model. However, without a concrete, measurable set of requirements any adaptations we make to the user model may possibly be misguided and perhaps even detrimental.

Our correction model, based on a multi-agent bidding process and the aforementioned metrics and utility function, takes advantage of multiple agents to suggest corrections to the interface agent's user model. Our analysis and preliminary results indicate we can delegate the maintenance of the interface agent's user model to other agents. The benefits are twofold: we can extend the interface agent's existing adaptation mechanisms by using the correction adaptation agents to correct the user model and the interface agent does not use additional computational resources determining how to modify the user model. Based on our particular methodology for constructing user models, we identified problems that may occur in the model [6] and have implemented several simple, but useful correction adaptation agents. However, there is a larger issue here. For any given user model, the correction adaptation agents we have currently implemented may not be sufficient to correct the user model. As mentioned previously, certain correction adaptation agents are more suited to improve certain metrics. We could attempt to build several (many) more correction adaptation agents capable of different adaptations. However, which ones do we build? What sort of adaptations to the interface agent user model are really needed? Do we have the right "mix" of correction adaptation agents? That is, to use multi-agent systems' terminology, how *diverse* is our collection of correction adaptation agents? Other researchers have explored the concept of *behavioral diversity* within multi-agent teams [1, 10, 15], indicating this sort of diversity is beneficial for some tasks. While although it is hypothetically possible to construct more agents, we desire to take a top-down approach to the construction of correction adaptation agents. Therefore, we desire to be able to decompose the correction adaptation agents into their atomic parts, determining first how the agents are composed. Given this decomposition, we then desire to be able to combine various parts together to make new agents. This is an active, on-going research topic.

Future efforts propose to provide tools to developers for constructing interface agent user models. Current agent development environments focus on the collaboration and autonomy requirements of an agent (more the former than the latter), while ignoring the adaptivity and robustness requirements. We propose to address these issues explicitly within our development environment, while additionally concentrating on environment specification and agent knowledge base and reasoning mechanisms.

References

[1] Tucker Balch. Learning roles: Behavioral diversity in robot teams. In *Collected Papers from the 1997 AAAI Workshop on Multiagent Learning*, pages 7–12. AAAI Press, July 1997. AAAI Technical Report WS-97-03.

[2] Sheila B. Banks, Robert A. Harrington, Eugene Santos Jr., and Scott M. Brown. Usability testing of an intelligent interface agent. In *Proceedings of the Sixth International Interfaces Conference (Interfaces 97)*, pages 121–123, May 1997.

[3] Sheila B. Banks, Martin R. Stytz, Eugene Santos Jr., and Scott M. Brown. User modeling for military training: Intelligent interface agents. In *Proceedings of the 19th Interservice/Industry Training Systems and Education Conference*, pages 645–653, December 1997.

[4] David Bawcom. An incompleteness handling methodology for validation of bayesian knowledge bases. Master's thesis, Air Force Institute of Technology, 1997.

[5] Scott M. Brown, Robert A. Harrington, Eugene Santos Jr., and Sheila B. Banks. User models, interface agents and expert systems. In *Proceedings of the Embedding User Models in Intelligent Applications Workshop*, pages 12–17, June 1997. held in conjunction with the Sixth International Conference on User Modeling (UM '97).

[6] Scott M. Brown, Eugene Santos Jr., and Sheila B. Banks. A dynamic bayesian intelligent interface agent. In *Proceedings of the Sixth International Interfaces Conference (Interfaces 97)*, pages 118–120, May 1997.

[7] Scott M. Brown, Eugene Santos Jr., and Sheila B. Banks. Utility theory-based user models for intelligent interface agents. In *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence (AI '98)*, June 1998. to appear.

[8] K. S. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In W. L. Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 404–413, Marina del Rey, February 1997.

[9] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, 1996.

[10] Dani Goldberg and Maja J Matarić. Interference as a tool for designing and evaluating multi-robot controllers. In *Proceedings of the Fourteenth National Conference on Artifical Intelligence (AAAI 97)*, pages 637–642, July 1997.

[11] Richard Goodwin. Formalizing properties of agents. Technical Report CMU-CS-93-159, Carnegie Mellon Institute, 1993.

[12] Robert A. Harrington, Sheila Banks, and Eugene Santos Jr. Development of an intelligent user interface for a generic expert system. In Michael Gasser, editor, *Online Proceedings of the Seventh Midwest Artificial Intelligence and Cognitive Science Conference*, 1996. Available at http://www.cs.indiana.edu/event/maics96/.

[13] Robert A. Harrington, Sheila Banks, and Eugene Santos Jr. GESIA: Uncertainty-based reasoning for a generic expert system intelligent user interface. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence*, pages 52–55, 1996.

[14] Robert A. Harrington and Scott M. Brown. Intelligent interface learning with uncertainty. In Eugene Santos Jr., editor, *Proceedings of the Eighth Midwest Artificial Intelligence and Cognitive Science Conference*, pages 27–34. AAAI Press, 1997.

[15] Thomas Haynes and Sandip Sen. Crossover operators for evolving a team. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*. MIT Press, 1997.

[16] Eric Horvitz and Matthew Barry. Display of information for time-critical decision making. In *Proceedings of the Eleventh Uncertainty in Artificial Intelligence*, pages 296–305, 1995.

[17] Anthony Jameson. Numeric uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interactions*, 5:193–251, 1995.

[18] Anthony Jameson, Cécil Paris, and Carlo Tasso, editors. *Preface of User Modeling: Proceedings of the Sixth International Conference, UM97*. Springer Wien New York, 1997.

[19] Patti Maes. Modeling adaptive autonomous agents. *Artificial Life Journal*, 1(1 & 2), 1994. MIT Press (C. Langton, Ed.).

[20] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):811–821, July 1994.

[21] James Mayfield, Yannis Labrou, and Tim Finin. Evaluation of kqml as an agent communication language. In Michael J. Woolridge, Jörg P. Müller, and Milind Tambe, editors, *Intelligent Agents II: Agent Theories, Architectures, and Languages*, pages 347–360. Berlin: Springer, 1996.

[22] Jörg P. Müller. A cooperation model for autonomous agents. In Jörg P. Müller, Michael J. Woolridge, and Nicholas R. Jennings, editors, *Intelligent Agents III: Agent Theories, Architectures, and Languages*, ECAI'96 Workshop (ATAL), pages 245–260. Springer, August 1996.

[23] Douglas W. Oard and Gary Marchionini. A conceptual framework for text filtering. Technical Report CS-TR-3643, University of Maryland, College Park, MD, May 1996.

[24] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[25] Charles Petrie. Agent-based engineering, the web, and intelligence. *IEEE Expert*, 11(6):24–29, December 1996.

[26] Elaine Rich. Users are individuals: Individualizing user models. *International Journal of Man-Machine Studies*, 18:199–214, 1983.

[27] Eugene Santos Jr., Darwyn O. Banks, and Sheila B. Banks. MACK: A tool for acquiring consistent knowledge under uncertainty. In *Proceedings of the AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, pages 23–32, 1997.

[28] Eugene Santos Jr., Howard T. Gleason, and Sheila B. Banks. BVAL: Probabilistic knowledge-base validation. In *Proceedings of the AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, pages 13–22, 1997.

[29] Solomon Eyal Shimony, Carmel Domshlak, and Eugene Santos Jr. Cost-sharing heuristic for bayesian knowledge-bases. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 421–428, 1997.

[30] John D. Spragins, Joseph L. Hammond, and Krzysztof Pawlikowski. *Telecommunications: Protocols and Design*. Addison Wesley, 1991.

[31] Daniel J. Stein III, Sheila B. Banks, Eugene Santos Jr., and Michael L. Talbert. Utilizing goal-directed data mining for incompleteness repair in knowledge bases. In Eugene Santos Jr., editor, *Proceedings of the Eighth Midwest Artificial Intelligence and Cognitive Science Conference*, pages 82–85. AAAI Press, 1997.

[32] Katia Sycara, Keith Decker, Anandeep Pannu, Mike Williamson, and Dajun Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6):36–46, December 1996.

[33] David Zeng and Katia Sycara. How can an agent learn to negotiate? In Jörg P. Müller, Michael J. Woolridge, and Nicholas R. Jennings, editors, *Intelligent Agents III: Agent Theories, Architectures, and Languages*, ECAI'96 Workshop (ATAL), pages 233–244. Springer, August 1996.